AMENDMENTS TO THE CLAIMS

Please amend claims 1-24 as shown below.

1. (Currently amended) In a system in which a hardware target computer system, which has a target instruction set architecture (ISA), executes a target instruction sequence corresponding to a source instruction sequence of a source system, which has a source ISA and is running on the target computer system, a method for handling exceptions comprising the following steps:

converting the source instruction sequence into the target instruction sequence by binary translation, each instruction in the source instruction sequence being converted into a corresponding translated target instruction sequence, which may consist of a single target instruction;

determining beginning and ending addresses of each source instruction and each corresponding translated target instruction;

generating a mapping between the beginning and ending addresses of each source instruction and its corresponding translated target instruction sequence;

executing the translated target instruction sequence;

at any point in the translated target instruction sequence, sensing the presence of an exception and determining whether each sensed exception is synchronous or asynchronous,

a synchronous exception being defined as an exception resulting from attempted execution of a target instruction, in which synchronous exceptions are of either of two types, namely, transparent and non-transparent, a transparent exception being defined as an exception requiring processing action wholly within the target computer system, and a non-transparent exception being defined as an exception requiring processing that alters a visible state of the source system; and

an asynchronous exception being defined as an exception resulting from an event unrelated to the execution of a target instruction; ~~and~~

if the sensed exception is asynchronous, resuming to completion the execution of the target instruction sequence in binary translation from the point in the translated target instruction sequence at which the asynchronous exception was sensed before handling the asynchronous exception and thus delaying handling of each

asynchronous sensed exception until, and no later than, completion of execution of the target instruction sequence corresponding to the current source instruction when the asynchronous exception is sensed;

determining whether the sensed synchronous exception is transparent or non-transparent;

handling each transparent synchronous exception externally from the source system, the visible state of the source system thereby remaining unaltered; and

forwarding to the source system for processing each non-transparent synchronous exception.

Claims 2-5 (Canceled).

6. (Currently amended) A method as in claim ~~5~~ 1, in which the step of forwarding each non-transparent synchronous exception to the source system includes the step of converting the sensed exception into a simulated source exception in a source instruction stream, which is sensed by and interrupts the source system.

7. (Canceled)

8. (Currently amended) A method as in claim ~~7~~ 1, further including the following steps:

determining a source instruction pointer as a predetermined function of the final target instruction pointer;

forwarding and processing the sensed asynchronous exception;

resuming execution at the location in the translation cache that corresponds to a current source instruction pointer; and

asynchronous exceptions thereby being processed only upon completion of execution of the translated target instructions corresponding to whole source instructions.

9. (Original) A method as in claim 8, in which the step of delaying processing of the sensed asynchronous exception further includes the step of simulating execution of the remaining target instructions.

10. (Original) A method as in claim 8, in which the step of delaying processing of the sensed asynchronous exception further includes the step of single-stepping the execution of the remaining target instructions.

11. (Original) A method as in claim 8, in which the step of delaying processing of the sensed asynchronous exception further includes the following steps:

temporarily replacing with a trap generation instruction the initial target instructions in each of the translated target instruction sequences that correspond to target instruction sequences that possibly immediately follow the current target instruction sequence;

resuming execution of the current target instruction sequence from the point at which the asynchronous exception was sensed;

restoring each of the temporarily replaced instructions with their original content after completion of the processing of the sensed asynchronous exception; and

upon reaching the trap generation instruction, forwarding and processing the sensed asynchronous exception.

12. (Original) A method as in claim 8, in which the step of delaying the processing of the sensed asynchronous exception further includes the following steps:

temporarily replacing with a trap generation instruction each indirect branch instruction, each indirect branch instruction corresponding to a possible last instruction of the current target instruction sequence;

resuming execution of the current target instruction sequence from the point at which the asynchronous exception was processed;

restoring each of the temporarily replaced instructions with their original content; and

simulating the restored indirect branch instruction.

13. (Original) A method as in claim 1, in which:

the source system is a virtual machine;

a virtual machine monitor that is operationally installed between the virtual machine and the hardware target computer system, the virtual machine thereby running on the virtual machine monitor;

the steps of converting the source instruction sequence into the target instruction sequence by binary translation, executing the translated target instruction sequence, sensing the presence of an exception, and delaying application of the sensed exception, are carried out by the virtual machine monitor.

14. (Original) A method as in claim 1, in which the source ISA is identical to the target ISA.

15. (Previously presented) In a system in which a hardware target computer system, which has a target instruction set architecture (ISA), executes a target instruction sequence corresponding to a source instruction sequence of a source system, which has a source ISA and is running on the target computer system, a method for handling exceptions comprising the following steps:

converting the source instruction sequence into the target instruction sequence by binary translation, each instruction in the source instruction sequence being converted into a corresponding translated target instruction sequence, which may consist of a single target instruction;

executing the translated target instruction sequence;

at any point in the translated target instruction sequence, sensing the presence of an exception,

each exception being of either of two types -- synchronous and asynchronous – a synchronous exception being defined as an exception resulting from attempted execution of a target instruction and an asynchronous exception is defined as an exception resulting from an event unrelated to the execution of a target instruction,

synchronous exceptions being of either of two types, namely, transparent and non-transparent, a transparent exception being defined as an exception requiring processing action wholly within the target computer system, and a non-transparent exception being defined as an exception requiring processing that alters a visible state of the source system;

determining whether each sensed exception is synchronous or asynchronous;

determining whether each sensed synchronous exception is transparent or non-transparent;

upon sensing the presence of an asynchronous exception during execution of a current one of the translated target instruction sequences, delaying processing of the sensed asynchronous exception until completion of the remaining target instructions in the current translated target instruction sequence;

determining beginning and ending addresses of each source instruction and each corresponding translated target instruction;

generating a mapping between the beginning and ending addresses of each source instruction and its corresponding translated target instruction sequence;

handling each transparent synchronous exception externally from the source system, the visible state of the source system thereby remaining unaltered;

forwarding to the source system for processing each non-transparent synchronous exception;

determining a source instruction pointer as a predetermined function of the final target instruction pointer;

forwarding and processing each sensed asynchronous exception;

resuming execution at the location in the translation cache that corresponds to a current source instruction pointer, that is, from the point in the translated target instruction sequence at which the asynchronous exception was sensed, asynchronous exceptions thereby being processed only upon completion of execution of the translated target instructions corresponding to whole source instructions;

delaying handling of each asynchronous sensed exception until, and no later than, completion of execution of the target instruction sequence corresponding to the current source instruction when the asynchronous exception is sensed;

in which:

the source system is a virtual machine;

a virtual machine monitor that is operationally installed between the virtual machine and the hardware target computer system, the virtual machine thereby running on the virtual machine monitor; and

the steps of converting the source instruction sequence into the target instruction sequence by binary translation, executing the translated target instruction sequence, sensing the presence of an exception, and delaying application of the sensed exception, are carried out by the virtual machine monitor.

16. (Currently amended) A system for virtualizing a computer system using binary translation comprising:

a hardware target computer system that has a target instruction set architecture (ISA) and executes a target instruction sequence;

a source system that has a source ISA and a source instruction sequence;

an intermediate software layer forming an interface between the source system and the hardware target computer system;

a binary translator included within the intermediate software layer and converting the source instruction sequence into the target instruction sequence by binary translation, each instruction in the source instruction sequence being converted into a corresponding translated target instruction sequence, which may consist of a single target instruction;

an exception handler included within the intermediate software layer and, at any point in the translated target instruction sequence, sensing the presence of an exception and determining whether the exception is synchronous or asynchronous, in which

an asynchronous exception is defined as an exception resulting from an event unrelated to the execution of a target instruction;

a synchronous exception is defined as an exception resulting from attempted execution of a target instruction and is of either of two types, namely, transparent and non-transparent, a transparent exception being defined as an exception requiring processing action wholly within the target computer system, and a non-transparent exception being defined as an exception requiring processing that alters a visible state of the source system;

a resume component within the intermediate software layer for resuming to completion the execution of the target instruction sequence in binary translation from the point in the translated target instruction sequence at which the asynchronous exception was sensed if the sensed exception is asynchronous, before handling of the asynchronous exception and

the exception handler further delaying handling of each asynchronous sensed exception until, and no later than, completion of execution of the target instruction

sequence corresponding to the current source instruction when the asynchronous exception is sensed;

in which the exception handler is further provided:

for determining whether the sensed synchronous exception is transparent or non-transparent;

for handling each transparent synchronous exception externally from the source system, the visible state of the source system thereby remaining unaltered; and

for forwarding to the source system for processing each non-transparent synchronous exception.

17. (Previously presented) A system as in claim 16, further including a map containing a mapping between beginning and ending addresses of each source instruction and its corresponding translated target instruction sequence.

18. (Canceled)

19. (Canceled)

20. (Currently amended) A system as in claim ~~19~~ 16, in which the exception handler is further provided for converting the sensed exception into a simulated source exception in a source instruction stream, which is sensed by and interrupts the source system.

21. (Currently amended) A system as in claim ~~19~~ 16, in which the exception handler is further provided, upon sensing the presence of an asynchronous exception during execution of a current one of the translated target instruction sequences, for delaying processing of the sensed asynchronous exception until completion of the remaining target instructions in the current translated target instruction sequence.

22. (Currently amended) A system as in claim ~~19~~ 16, in which the exception handler is further provided

for determining a source instruction pointer as a predetermined function of the final target instruction pointer;

for forwarding and processing the sensed asynchronous exception; and

for resuming execution at the location in the translation cache that corresponds to a current source instruction pointer, asynchronous exceptions thereby being processed only upon completion of execution of the translated target instructions corresponding to whole source instructions.

23. (Previously presented) A system as in claim 16, in which:

the source system is a virtual machine;

the system further includes a virtual machine monitor that is operationally installed between the virtual machine and the hardware target computer system, the virtual machine thereby running on the virtual machine monitor; and

the binary translator and exception handler are included within the virtual machine monitor.

24. (Original) A method as in claim 16, in which the source ISA is identical to the target ISA.